Intel® Xeon™ Processor with 800 MHz System Bus Specification Update

August 2004

Notice: The Intel® Xeon™ Processor with 800 MHz System Bus may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are documented in this specification update.

Document Number: 302402-003

Contents

Revision History

Version	Description	Date
-001	Initial release of the document.	July 2004
-002	Removed erratum P18 and renumbered existing errata.	July 2004
-003	Added errata S32 – S34. Renamed errata numbering from P to S.	August 2004

Preface

This document is an update to the specifications contained in the following documents:

1. *Intel® Xeon™ Processor with 800 MHz System Bus Datasheet* (Document Number 302355)
2. *IA-32 Intel® Architecture Software Developer's Manual*, Volume 1: Basic Architecture (Document Number 253665)
3. *IA-32 Intel® Architecture Software Developer's Manual*, Volume 2A: Instruction Set Reference, A-M (Document Number 253666)
4. *IA-32 Intel® Architecture Software Developer's Manual*, Volume 2B: Instruction Set Reference, N-Z (Document Number 253667)
5. *IA-32 Intel® Architecture Software Developer's Manual*, Volume 3: System Programming Guide (Document Number 253668)
6. *Intel® Extended Memory 64 Technology Software Developer's Guide*, Volume 1 (Document Number 300834)
   Here is the link: http://developer.intel.com/technology/64bitextensions/300834.htm
7. *Intel® Extended Memory 64 Technology Software Developer's Guide*, Volume 2 (Document Number 300835)
   Here is the link: http://developer.intel.com/technology/64bitextensions/300835.htm.

This document is intended for hardware system manufacturers and software developers of applications, operating systems, or tools.

Nomenclature

S-Spec Number is a five-digit code used to identify products. Products are differentiated by their unique characteristics, e.g., core speed, L2 cache size, package type, etc. as described in the processor identification information table. Care should be taken to read all notes associated with each S-Spec number.

Errata are design defects or errors. Errata may cause the processor's behavior to deviate from published specifications. Hardware and software designed to be used with any given processor must assume that all errata documented for that processor are present on all devices unless otherwise noted.

Specification Changes are modifications to the current published specifications. These changes will be incorporated in the next release of the specifications.

Specification Clarifications describe a specification in greater detail or further highlight a specification's impact to a complex design situation. These clarifications will be incorporated in the next release of the specifications.

Documentation Changes include typos, errors, or omissions from the current published specifications. These changes will be incorporated in the next release of the specifications.

Identification Information

Intel® Xeon™ Processor with 800 MHz System Bus Package Markings (604-pin FC-mPGA4 Package)

Figure 1. Top Side Processor Marking Example

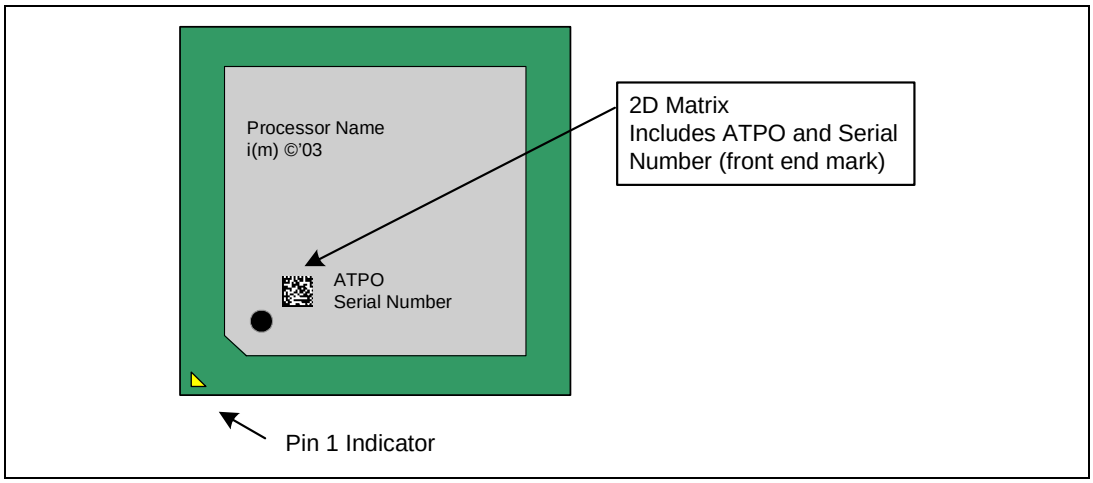
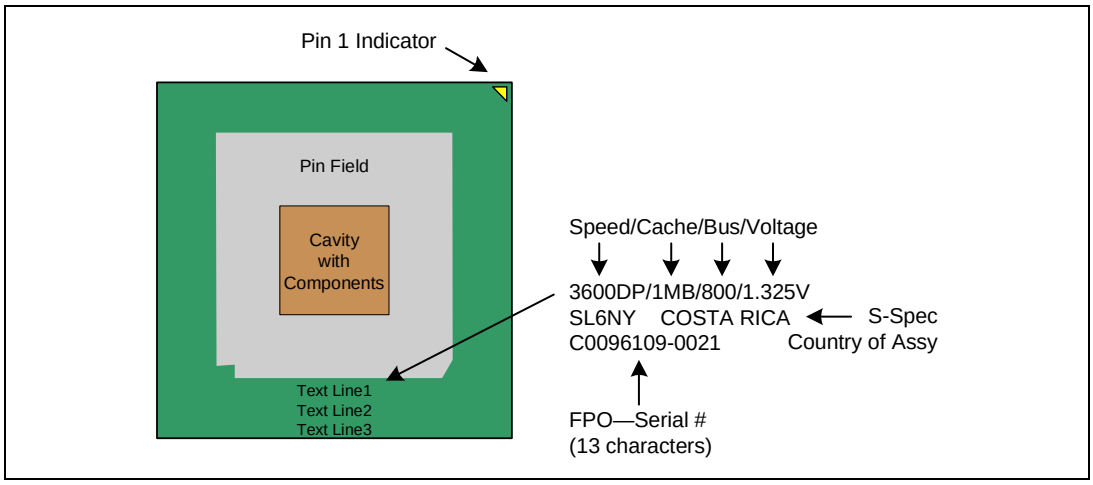


Figure 2. Bottom Side Processor Marking Example



The Intel® Xeon™ Processor with 800 MHz System Bus can be identified by the following values:

Table 1. Identification Information

Family[1]	Model[2]	Brand ID[3]
1111b	0100b	0000b

NOTES:

1. The Family corresponds to bits [11:8] of the EDX register after RESET, bits [11:8] of the EAX register after the CPUID instruction is executed with a 1 in the EAX register, and the generation field of the Device ID registers accessible through Boundary Scan.
2. The Model corresponds to bits [7:4] of the EDX register after RESET, bits [7:4] of the EAX register after the CPUID instruction is executed with a 1 in the EAX register, and the model field of the Device ID registers accessible through Boundary Scan.
3. Brand ID returns 0000b, which means that Brand ID is unsupported in this processor.

Cache and TLB descriptor parameters are provided in the EAX, EBX, ECX and EDX registers after the CPUID instruction is executed with a 2 in the EAX register. Please refer to the *Intel Processor Identification and the CPUID Instruction Application Note* (AP-485) for further information on the CPUID instruction.

Table 2. Intel® Xeon™ Processor with 800 MHz System Bus Identification Information

QDF1/ S-Spec	Core Stepping	CPUID	Core Freq (GHz)	Data Bus Freq (MHz)	L2 Cache Size	Processor Package Revision	Package and Revision	Notes
SL7DV	D-0	0F34h	2.80	800	1 MB	01	604-pin micro-PGA with 42.5 x 42.5 mm FC-PGA4 package	1, 2, 3
SL7HF								1, 2, 3, 4
SL7DW	D-0	0F34h	3	800	1 MB	01	604-pin micro-PGA with 42.5 x 42.5 mm FC-PGA4 package	1, 2, 3
SL7HG								1, 2, 3, 4
SL7DX	D-0	0F34h	3.20	800	1 MB	01	604-pin micro-PGA with 42.5 x 42.5 mm FC-PGA4 package	1, 2, 3
SL7HH								1, 2, 3, 4
SL7DY	D-0	0F34h	3.40	800	1 MB	01	604-pin micro-PGA with 42.5 x 42.5 mm FC-PGA4 package	1, 2, 3
SL7HJ								1, 2, 3, 4
SL7DZ	D-0	0F34h	3.60	800	1 MB	01	604-pin micro-PGA with 42.5 x 42.5 mm FC-PGA4 package	1, 2, 3
SL7HK								1, 2, 3, 4

NOTES:

1. These are production parts.
2. These parts are enabled for Intel® Extended Memory 64 Technology.
3. These parts have Tcontrol programmed.
4. These are Intel boxed processors.

Mixed Steppings in DP Systems

Intel Corporation fully supports mixed steppings of Intel Xeon Processor with 800 MHz System Bus. The following list and processor matrix describes the requirements to support mixed steppings:

- Mixed steppings are only supported with processors that have identical family numbers as indicated by the `CPUID` instruction.
- While Intel has done nothing to specifically prevent processors operating at differing frequencies from functioning within a multiprocessor system, there may be uncharacterized errata that exist in such configurations. Intel does not support such configurations. In mixed stepping systems, all processors must operate at identical frequencies (i.e., the highest frequency rating commonly supported by all processors).
- While there are no known issues associated with the mixing of processors with differing cache sizes in a multiprocessor system, and Intel has done nothing to specifically prevent such system configurations from operating, Intel does not support such configurations since there may be uncharacterized errata that exist. In mixed stepping systems, all processors must be of the same cache size.
- While Intel believes that certain customers may wish to perform validation of system configurations with mixed frequency or cache sizes, and that those efforts are an acceptable option to our customers, customers would be fully responsible for the validation of such configurations.
- Intel requires that the proper microcode update be loaded on each processor operating in a multiprocessor system. Any processor that does not have the proper microcode update loaded is considered by Intel to be operating out of specification.
- The workarounds identified in this and following specification updates must be properly applied to each processor in the system. Certain errata are specific to the multiprocessor environment. Errata for all processor steppings will affect system performance if not properly worked around. Also see Table 2 for additional details on which processors are affected by specific errata.
- In mixed stepping systems, the processor with the lowest feature-set, as determined by the `CPUID` Feature Bytes, must be the Bootstrap Processor (BSP). In the event of a tie in feature-set, the tie should be resolved by selecting the BSP as the processor with the lowest stepping as determined by the `CPUID` instruction.

Summary Table of Changes

The following table indicates the Errata, Documentation Changes, Specification Clarifications, or Specification Changes that that apply to Intel processors. Intel intends to fix some of the errata in a future stepping of the component, and to account for the other outstanding issues through documentation or specification changes as noted. This table uses the following notation:

Codes Used In Summary Table

X:	Specification Change, Erratum, Specification Clarification, or Documentation Change applies to the given processor stepping.
(No mark) or (blank box):	This item is fixed in or does not apply to the given stepping.
Doc:	Document change or update that will be implemented in a future revision.
Plan Fix:	This erratum may be fixed in a future stepping of the product.
Fixed:	This erratum has been previously fixed.
No Fix:	There are no plans to fix this erratum.
	Change bar to left of table row indicates this item is either new or modified from the previous version of the document.

Each Specification Update item will be prefixed with a capital letter to distinguish the product. The key below details the letters that are used in Intel's microprocessor Specification Updates:

A = Intel® Pentium®II processor

B = Mobile Intel® Pentium®II processor

C = Intel® Celeron® processor

D = Intel® Pentium® II Xeon™ processor

E = Intel® Pentium® III processor

G = Intel® Pentium® III Xeon™ processor

H = Mobile Intel® Celeron® processor at 466 MHz, 433 MHz, 400 MHz, 366 MHz, 333 MHz, 300 MHz, and 266 MHz

K = Mobile Intel® Pentium® III processor - M

L = Unannounced Intel® Celeron® D processor

M = Mobile Intel® Celeron® processor

N = Intel® Pentium® 4 processor

O = Intel® Xeon™ processor MP

P = Intel® Xeon™ processor

R = Intel® Pentium® 4 processor on 90 nm process

S = Intel® Xeon™ Processor with 800 MHz System Bus

T = Mobile Intel® Pentium® 4 processor

U = Unannounced Gallatin processor

V = Mobile Intel® Celeron® processor on.13 Micron Process in Micro-FCPGA Package

X = Intel® Pentium® M processor on 90nm process with 2-MB L2 Cache

Y = Intel® Pentium® M processor

Z = Unannounced Mobile Intel® Pentium® 4 processor with 533 MHz system bus

The Specification Updates for the Pentium® processor, Pentium® Pro processor, and other Intel products do not use this convention.

Errata (Sheet 1 of 2)

No.	D-0/ 0F34h	Plans	Errata
S1	X	No Fix	Transaction is not retired after BINIT#
S2	X	No Fix	Invalid opcode 0FFFh requires a ModRM byte
S3	X	No Fix	Processor may hang due to speculative page walks to non-existent system memory
S4	X	No Fix	Memory type of the load lock different from its corresponding store unlock
S5	X	No Fix	Machine check architecture error reporting and recovery may not work as expected
S6	X	No Fix	Debug mechanisms may not function as expected
S7	X	No Fix	Cascading of performance counters does not work correctly when forced overflow is enabled
S8	X	No Fix	EMON event counting of x87 loads may not work as expected
S9	X	No Fix	System bus interrupt messages without data and which receive a hard-failure response may hang the processor
S10	X	No Fix	The processor signals page-fault exception (#PF) instead of alignment check exception (#AC) on an unlocked CMPXCHG8B instruction
S11	X	No Fix	FSW may not be completely restored after page fault on FRSTOR or FLDENV instructions
S12	X	No Fix	Processor issues inconsistent transaction size attributes for locked operation
S13	X	No Fix	When the processor is in the system management mode (SMM), Debug registers may be fully writeable
S14	X	No Fix	Shutdown and IERR# may result due to a machine check exception on a Hyper-Threading Technology enabled processor
S15	X	No Fix	Processor may hang under certain frequencies and 12.5% STPCLK# duty cycle
S16	X	No Fix	System may hang if a fatal cache error causes bus write line (BWL) transaction to occur to the same cache line address as an outstanding bus read line (BRL) or bus read-invalidate line (BRIL)
S17	X	No Fix	A write to APIC task priority register (TPR) that lowers priority may seem to have not occurred

Errata (Sheet 2 of 2)

No.	D-0/ 0F34h	Plans	Errata
S18	X	No Fix	Parity error in the L1 cache may cause the processor to hang
S19		No Fix	Sequence of locked operations can cause two threads to receive stale data and cause application hang
S20	X	Plan Fix	A 16-bit address wrap resulting from a near branch (jump or call) may cause an incorrect address to be reported to the #GP exception handler
S21	X	No Fix	Bus locks and SMC detection may cause the processor to temporarily hang
S22	X	Plan Fix	Incorrect physical address size returned by CPUID instruction
S23	X	No Fix	Incorrect debug exception (#DB) may occur when a data breakpoint is set on an FP instruction
S24	X	No Fix	xAPIC may not report some illegal vector errors
S25	X	Plan Fix	Enabling no-eviction mode (NEM) may prevent the operation of the second logical processor in a Hyper-Threading Technology enabled boot strap processor (BSP)
S26	X	Plan Fix	Task priority register (TPR) updates during thermal monitor 2 (TM2), enhanced halt state (CIE), or enhanced Intel SpeedStep® technology (EIST) voltage transitions may cause system hang in DP capable platforms
S27	X	Plan Fix	Interactions between the instruction translation lookaside buffer (ITLB) and the instruction streaming buffer may cause unpredictable software behavior
S28	X	Plan Fix	STPCLK# signal assertion under certain conditions may cause a system hang
S29	X	No Fix	Missing Stop Grant Acknowledge special bus cycle may cause a system hang
S30	X	Plan Fix	Incorrect duty cycle is chosen when on-demand clock modulation is enabled in a processor supporting Hyper-Threading Technology
S31	X	No Fix	Memory aliasing of pages as uncacheable memory type and write back (WB) may hang the system
S32	X	No Fix	Using STPCLK# and executing code from very slow memory could lead to a system hang
S33	X	No Fix	Processor provides a 4-byte store unlock after an 8-byte load lock
S34	X	No Fix	Machine Check Architecture error reporting and recovery may not work as expected

Specification Changes

No.	Plans	SPECIFICATION CHANGES
		None for this revision of the Specification Update.

Specification Clarifications

No.	Plans	SPECIFICATION CLARIFICATIONS
		None for this revision of the Specification Update.

Documentation Changes

No.	Plans	DOCUMENTATION CHANGES
		None for this revision of the Specification Update.

Errata

S1 Transaction is not retired after BINIT#

Problem: If the first transaction of a locked sequence receives a HITM# and DEFER# during the snoop phase it should be retried and the locked sequence restarted. However, if BINIT# is also asserted during this transaction, the transaction will not be retried.

Implication: When this erratum occurs, locked transactions will not be retried.

Workaround: None at this time.

Status: For the steppings affected, see the *Summary Table of Changes*.

S2 Invalid opcode 0FFFh requires a ModRM byte

Problem: Some invalid opcodes require a ModRM byte and other following bytes, while others do not. The invalid opcode 0FFFh did not require a ModRM in previous generation microprocessors such as Pentium II or Pentium III processors, but it is required in the Intel Xeon processor.

Implication: The use of an invalid opcode 0FFFh without the ModRM byte may result in a page or limit fault on the Intel Xeon processor. When this erratum occurs, locked transactions will not be retried.

Workaround: To avoid this erratum use ModRM byte with invalid 0FFFh opcode.

Status: For the steppings affected, see the *Summary Table of Changes*.

S3 Processor may hang due to speculative page walks to non-existent system memory

Problem: A load operation issued speculatively by the processor that misses the data translation lookaside buffer (DTLB) results in a page walk. A branch instruction older than the load retires so that this load operation is now in the mispredicted branch path. Due to an internal boundary condition, in some instances the load is not canceled before the page walk is issued.

The page miss handler (PMH) starts a speculative page-walk for the Load and issues a cacheable load of the page directory entry (PDE). This PDE load returns data that points to a page table entry in uncacheable (UC) memory. The PMH issues the PTE Load to UC space, which is issued on the front side bus. No response comes back for this load PTE operation since the address is pointing to system memory, which does not exist.

This load to non-existent system memory causes the processor to hang because other bus requests are queued up behind this UC PTE load, which never gets a response. If the load was accessing valid system memory, the speculative page-walk would successfully complete and the processor would continue to make forward progress.

Implication: Processor may hang due to speculative page walks to non-existent system memory.

Workaround: Page directories and page tables in UC memory space must point to system memory that exists.

Status: For the steppings affected, see the *Summary Table of Changes*.

S4 Memory type of the load lock different from its corresponding store unlock

Problem: The Intel Xeon Processor employs a use-once protocol to ensure that a processor in a multiprocessor system may access data that is loaded into its cache on a read-for-ownership (RFO) operation at least once before it is snooped out by another processor. This protocol is necessary to avoid a dual processor livelock scenario where no processor in the system can gain ownership of a line and modify it before that data is snooped out by another processor. In the case of this erratum, the use-once protocol incorrectly activates for split load lock instructions. A load lock operation accesses data that splits across a page boundary with both pages of WB memory type. The use-once protocol activates and the memory type for the split halves get forced to UC. Since use-once does not apply to stores, the store unlock instructions go out as WB memory type. The full sequence on the Bus is: locked partial read (UC), partial read (UC), partial write (WB), locked partial write (WB). The Use-once protocol should not be applied to Load locks.

Implication: When this erratum occurs, the memory type of the load lock will be different than the memory type of the store unlock operation. This behavior (Load Locks and Store Unlocks having different memory types) does not however introduce any functional failures such as system hangs or memory corruption.

Workaround: None at this time.

Status: For the steppings affected, see the *Summary Table of Changes*.

S5 Machine check architecture error reporting and recovery may not work as expected

Problem: When the processor detects errors it should attempt to report and/or recover from the error. In the situations described below, the processor does not report and/or recover from the error(s) as intended.

- When a transaction is deferred during the snoop phase and subsequently receives a Hard Failure response, the transaction should be removed from the bus queue so that the processor may proceed. Instead, the transaction is not properly removed from the bus queue, the bus queue is blocked, and the processor will hang.
- When a hardware prefetch results in an uncorrectable tag error in the L2 cache, MC0_STATUS.UNCOR and MC0_STATUS.PCC are set but no machine check exception (MCE) is signaled. No data loss or corruption occurs because the data being prefetched has not been used. If the data location with the uncorrectable tag error is subsequently accessed, an MCE will occur. However, upon this MCE, or any other subsequent MCE, the information for that error will not be logged because MC0_STATUS.UNCOR has already been set and the MCA status registers will not contain information about the error which caused the MCE assertion but instead will contain information about the prefetch error event.
- When the reporting of errors is disabled for machine check architecture (MCA) Bank 2 by setting all MC2_CTL register bits to 0, uncorrectable errors should be logged in the IA32_MC2_STATUS register but no machine-check exception should be generated. Uncorrectable loads on bank 2, which would normally be logged in the IA32_MC2_STATUS register, are not logged.
- When one half of a 64 byte instruction fetch from the L2 cache has an uncorrectable error and the other 32 byte half of the same fetch from the L2 cache has a correctable error, the processor will attempt to correct the correctable error but cannot proceed due to the uncorrectable error. When this occurs the processor will hang.
- When an L1 cache parity error occurs, the cache controller logic should write the physical address of the data memory location that produced that error into the IA32_MC1_ADDR

REGISTER (MC1_ADDR). In some instances of a parity error on a load operation that hits the L1 cache, however, the cache controller logic may write the physical address from a subsequent load or store operation into the IA32_MC1_ADDR register.

- When an error exists in the tag field of a cache line such that a request for ownership (RFO) issued by the processor hits multiple tag fields in the L2 cache (the correct tag and the tag with the error) and the accessed data also has a correctable error, the processor will correctly log the multiple tag match error but will hang when attempting to execute the MCE handler.
- If a memory access receives a machine check error on both 64 byte halves of a 128-byte L2 cache sector, the IA32_MC0_STATUS register records this event as multiple errors, i.e., the valid error bit and the overflow error bit are both set indicating that a machine check error occurred while the results of a previous error were in the error-reporting bank. The IA32_MC1_STATUS register should also record this event as multiple errors but instead records this event as only one correctable error.
- The overflow bit should be set to indicate when more than one error has occurred. The overflow bit being set indicates that more than one error has occurred. Because of this erratum, if any further errors occur, the MCA overflow bit will not be updated; thereby incorrectly indicating only one error has been received.
- If an I/O instruction (IN, INS, REP INS, OUT, OUTS, or REP OUTS) is being executed, and if the data for this instruction becomes corrupted, the processor will signal a MCE. If the instruction is directed at a device that is powered down, the processor may also receive an assertion of SMI#. Since MCEs have higher priority, the processor will call the MCE handler, and the SMI# assertion will remain pending. However, while attempting to execute the first instruction of the MCE handler, the SMI# will be recognized and the processor will attempt to execute the SMM handler. If the SMM handler is successfully completed, it will attempt to restart the I/O instruction, but will not have the correct machine state due to the call to the MCE handler. This can lead to failure of the restart and shutdown of the processor.
- If PWRGOOD is deasserted during a RESET# assertion causing internal glitches, the MCA registers may latch invalid information.
- If RESET# is asserted, then deasserted, and reasserted, before the processor has cleared the MCA registers, then the information in the MCA registers may not be reliable, regardless of the state or state transitions of PWRGOOD.
- If MCERR# is asserted by one processor and observed by another processor, the observing processor does not log the assertion of MCERR#. The MCE handler called upon assertion of MCERR# will not have any way to determine the cause of the MCE.
- The Overflow Error bit (bit 62) in the IA32_MC0_STATUS register indicates, when set, that a machine check error occurred while the results of a previous error were still in the error reporting bank (i.e. The Valid bit was set when the new error occurred). If an uncorrectable error is logged in the error-reporting bank and another error occurs, the overflow bit will not be set.
- A different mechanism than the rest of the register writes the MCA Error Code field of the IA32_MC0_STATUS register. For uncorrectable errors, the other fields in the IA32_MC0_STATUS register are only updated by the first error. Any further errors that are detected will update the MCA Error Code field without updating the rest of the register, thereby leaving the IA32_MC0_STATUS register with stale information.
- When a speculative load operation hits the L2 cache and receives a correctable error, the IA32_MC1_Status Register may be updated with incorrect information. The IA32_MC1_Status Register should not be updated for speculative loads.

- The processor should only log the address for L1 parity errors in the IA32_MC1_Status register if a valid address is available. If a valid address is not available, the Address Valid bit in the IA32_MC1_Status register should not be set. In instances where an L1 parity error occurs and the address is not available because the linear to physical address translation is not complete or an internal resource conflict has occurred, the Address Valid bit is incorrectly set.
- The processor may hang when an instruction code fetch receives a hard failure response from the front side bus. This occurs because the bus control logic does not return data to the core, leaving the processor empty. IA32_MC0_STATUS MSR does indicate that a hard fail response occurred.
- The processor may hang when the following events occur and the machine check exception is enabled, CR4.MCE=1. A processor that has it's STPCLK# pin asserted will internally enter the Stop Grant State and finally issue a Stop Grant Acknowledge special cycle to the bus. If an uncorrectable error is generated during the Stop Grant process it is possible for the Stop Grant special cycle to be issued to the bus before the processor vectors to the machine check handler. Once the chipset receives its last Stop Grant special cycle it is allowed to ignore any bus activity from the processors. As a result, processor accesses to the machine check handler may not be acknowledged, resulting in a processor hang.

Implication: The processor is unable to correctly report and/or recover from certain errors

Workaround: None at this time.

Status: For the steppings affected, see the *Summary Table of Changes*.

S6 Debug mechanisms may not function as expected

Problem: If the first transaction of a locked sequence receives a HITM# and DEFER# during the snoop phase it should be retried and the locked sequence restarted. However, if BINIT# is also asserted during this transaction, the transaction will not be Certain debug mechanisms may not function as expected on the processor. The cases are as follows:

- When the following conditions occur: 1) An FLD instruction signals a stack overflow or underflow, 2) the FLD instruction splits a page-boundary or a 64 byte cache line boundary, 3) the instruction matches a Debug Register on the high page or cache line respectively, and 4) the FLD has a stack fault and a memory fault on a split access, the processor will only signal the stack fault and the debug exception will not be taken.
- When a data breakpoint is set on the ninth and/or tenth byte(s) of a floating-point store using the Extended Real data type, and an unmasked floating-point exception occurs on the store, the break point will not be captured.
- When any instruction has multiple debug register matches, and any one of those debug registers is enabled in DR7, all of the matches should be reported in DR6 when the processor goes to the debug handler. This is not true during a REP instruction. As an example, during execution of a REP MOVSW instruction the first iteration a load matches DR0 and DR2 and sets DR6 as FFFF0FF5h. On a subsequent iteration of the instruction, a load matches only DR0. The DR6 register is expected to still contain FFFF0FF5h, but the processor will update DR6 to FFFF0FF1h.

A Data breakpoint that is set on a load to uncacheable memory may be ignored due to an internal segment register access conflict. In this case the system will continue to execute instructions, bypassing the intended breakpoint. Avoiding having instructions that access segment descriptor registers e.g. LGDT, LIDT close to the UC load, and avoiding serialized instructions before the UC load will reduce the occurrence of this erratum.

Implication: Certain debug mechanisms do not function as expected on the processor.

Workaround: None at this time.

Status: For the steppings affected, see the *Summary Table of Changes*.

S7 Cascading of performance counters does not work correctly when forced overflow is enabled

Problem: The performance counters are organized into pairs. When the CASCADE bit of the Counter Configuration Control Register (CCCR) is set, a counter that overflows will continue to count in the other counter of the pair. The FORCE_OVF bit forces the counters to overflow on every non-zero increment. When the FORCE_OVF bit is set, the counter overflow bit will be set but the counter no longer cascades.

Implication: The performance counters do not cascade when the FORCE_OVF bit is set.

Workaround: None at this time.

Status: For the steppings affected, see the *Summary Table of Changes*.

S8 EMON event counting of x87 loads may not work as expected

Problem: If a performance counter is set to count x87 loads and floating-point exceptions are unmasked, the FPU Operand (Data) Pointer (FDP) may become corrupted.

Implication: When this erratum occurs, FPU Operand (Data) Pointer (FDP) may become corrupted.

Workaround: This erratum will not occur with floating point exceptions masked. If floating-point exceptions are unmasked, then performance counting of x87 loads should be disabled.

Status: For the steppings affected, see the *Summary Table of Changes*.

S9 System bus interrupt messages without data and which receive a hard-failure response may hang the processor

Problem: When a system bus agent (processor or chipset) issues an interrupt transaction without data onto the system bus, and the transaction receives a hard-failure response, a potential processor hang can occur. The processor, which generates an inter-processor interrupt (IPI) that receives hard-failure response, will still log the MCA error event cause as hard-failure, even if the APIC causes a hang. Other processors, which are true targets of the IPI, will also hang on hard failure-without-data, but will not record an MCA hard-failure event as a cause. If a hard-failure response occurs on a system bus interrupt message with data, the APIC will complete the operation so as not to hang the processor.

Implication: The processor may hang

Workaround: None at this time.

Status: For the steppings affected, see the *Summary Table of Changes*.

S10 The processor signals page-fault exception (#PF) instead of alignment check exception (#AC) on an unlocked CMPXCHG8B instruction

Problem: If a page-fault exception (#PF) and alignment check exception (#AC) both occur for an unlocked `CMPXCHG8B` instruction, then #PF will be flagged.

Implication: Software that depends on the #AC before the #PF will be affected since #PF is signaled in this case.

Workaround: Remove the software's dependency on #AC having precedence over #PF. Alternately, correct the page fault in the page fault handler and then restart the faulting instruction.

Status: For the steppings affected, see the *Summary Table of Changes*.

S11 FSW may not be completely restored after page fault on FRSTOR or FLDENV instructions

Problem: If the FPU operating environment or FPU state (operating environment and register stack) being loaded by an `FLDENV` or `FRSTOR` instruction wraps around a 64-Kbyte or 4-Gbyte boundary and a page fault (#PF) or segment limit fault (#GP or #SS) occurs on the instruction near the wrap boundary, the upper byte of the FPU status word (FSW) might not be restored. If the fault handler does not restart program execution at the faulting instruction, stale data may exist in the FSW.

Implication: When this erratum occurs, stale data will exist in the FSW.

Workaround: Ensure that the FPU operating environment and FPU state do not cross 64-Kbyte or 4-Gbyte boundaries. Alternately, ensure that the page fault handler restarts program execution at the faulting instruction after correcting the paging problem.

Status: For the steppings affected, see the *Summary Table of Changes*.

S12 Processor issues inconsistent transaction size attributes for locked operation

Problem: When the processor is in the Page Address Extension (PAE) mode and detects the need to set the Access and/or Dirty bits in the page directory or page table entries, the processor sends an 8-byte load lock onto the system bus. A subsequent 8 byte store unlock is expected, but instead a 4 byte store unlock occurs. Correct data is provided since only the lower bytes change, however external logic monitoring the data transfer may be expecting an 8-byte store unlock.

Implication: This erratum affects no known commercially available chipsets.

Workaround: None at this time.

Status: For the steppings affected, see the *Summary Table of Changes*.

S13 When the processor is in the system management mode (SMM), Debug registers may be fully writeable

Problem: When in system management mode (SMM), the processor executes code and stores data in the SMRAM space. When the processor is in this mode and writes are made to DR6 and DR7, the processor should block writes to the reserved bit locations. Due to this erratum, the processor may not block these writes. This may result in invalid data in the reserved bit locations.

Implication: Reserved bit locations within DR6 and DR7 may become invalid.

Workaround: Software may perform a read/modify/write when writing to DR6 and DR7 to ensure that the values in the reserved bits are maintained.

Status: For the steppings affected, see the *Summary Table of Changes*.

S14 Shutdown and IERR# may result due to a machine check exception on a Hyper-Threading Technology enabled processor

Problem: When a MCE occurs due to an internal error, both logical processors on a Hyper-Threading (HT) Technology enabled processor normally vector to the MCE handler. However, if one of the logical processors is in the "Wait for SIPI" state, that logical processor will not have a MCE handler and will shut down and assert IERR#.

Implication: A processor with a logical processor in the "Wait for SIPI" state will shut down when an MCE occurs on the other thread.

Workaround: None at this time.

Status: For the steppings affected, see the *Summary Table of Changes*.

S15 Processor may hang under certain frequencies and 12.5% STPCLK# duty cycle

Problem: If a system deasserts STPCLK# at a 12.5% duty cycle, and the processor is running below 2 GHz, and the processor thermal control circuit (TCC) on-demand clock modulation is active, the processor may hang. This erratum does not occur under the automatic mode of the TCC.

Implication: When this erratum occurs, the processor will hang.

Workaround: If use of the on-demand mode of the processor's TCC is desired in conjunction with STPCLK# modulation, then assure that STPCLK# is not asserted at a 12.5% duty cycle.

Status: For the steppings affected, see the *Summary Table of Changes*.

S16 System may hang if a fatal cache error causes bus write line (BWL) transaction to occur to the same cache line address as an outstanding bus read line (BRL) or bus read-invalidate line (BRIL)

Problem: A processor internal cache fatal data ECC error may cause the processor to issue a bus write line (BWL) transaction to the same cache line address as an outstanding bus read line (BRL) or bus read-invalidate line (BRIL). As it is not typical behavior for a single processor to have a BWL and a BRL/BRIL concurrently outstanding to the same address, this may represent an unexpected scenario to system logic within the chipset.

Implication: The processor may not be able to fully execute the machine check handler in response to the fatal cache error if system logic does not ensure forward progress on the system bus under this scenario.

Workaround: System logic should ensure completion of the outstanding transactions. Note that during recovery from a fatal data ECC error, memory image coherency of the BWL with respect to BRL/BRIL transactions is not important. Forward progress is the primary requirement.

Status: For the steppings affected, see the *Summary Table of Changes*.

S17 A write to APIC task priority register (TPR) that lowers priority may seem to have not occurred

Problem: Uncacheable stores to the APIC space are handled in a non-synchronous way with respect to the speed at which instructions are retired. If an instruction that masks the interrupt flag (for example CLI) is executed soon after an uncacheable write to the task priority register (TPR) that lowers the APIC priority the interrupt masking operation may take effect before the actual priority has been lowered. This may cause interrupts whose priority is lower than the initial TPR but higher than the final TPR to not be serviced until the interrupt flag is finally cleared (for example STI). Interrupts will remain pended and are not lost

Implication: This condition may allow interrupts to be accepted by the processor but may delay their service

Workaround: This can be avoided by issuing a TPR Read after a TPR Write that lowers the TPR value. This will force the store to the APIC priority resolution logic before any subsequent instructions are executed. No commercial operating system is known to be impacted by this erratum.

Status: For the steppings affected, see the *Summary Table of Changes*.

S18 Parity error in the L1 cache may cause the processor to hang

Problem: If a locked operation accesses a line in the L1 cache that has a parity error, it is possible that the processor may hang while trying to evict the line.

Implication: If this erratum occurs, it may result in a system hang. Intel has not observed this erratum with any commercially available software.

Workaround: None at this time.

Status: For the steppings affected, see the *Summary Table of Changes*.

S19 Sequence of locked operations can cause two threads to receive stale data and cause application hang

Problem: While going through a sequence of locked operations, it is possible for the two threads to receive stale data. This is a violation of expected memory ordering rules and causes the application to hang.

Implication: When this erratum occurs in an Hyper-Thread Technology enabled system, an application may hang.

Workaround: It is possible for the BIOS to contain a workaround for this erratum.

Status: For the steppings affected, see the *Summary Table of Changes*.

S20 A 16-bit address wrap resulting from a near branch (jump or call) may cause an incorrect address to be reported to the #GP exception handler

Problem: If a 16-bit application executes a branch instruction that causes an address wrap to a target address outside of the code segment, the address of the branch instruction should be provided to the general protection exception handler. It is possible that, as a result of this erratum, that the general protection handler may be called with the address of the branch target.

Implication: A 16-bit software environment which is affected by this erratum, will see that the address reported by the exception handler points to the target of the branch, rather than the address of the branch instruction.

Workaround: None at this time.

Status: For the steppings affected, see the *Summary Table of Changes*.

S21 Bus locks and SMC detection may cause the processor to temporarily hang

Problem: The processor may temporarily hang in an HT Technology enabled system, if one logical processor executes a synchronization loop that includes one or more bus locks and is waiting for release by the other logical processor. If the releasing logical processor is executing instructions that are within the detection range of the self-modifying code (SMC) logic, then the processor may be locked in the synchronization loop until the arrival of an interrupt or other event.

Implication: If this erratum occurs in an HT Technology enabled system, the application may temporarily stop making forward progress. Intel has not observed this erratum with any commercially available software.

Workaround: None at this time.

Status: For the steppings affected, see the *Summary Table of Changes*.

S22 Incorrect physical address size returned by CPUID instruction

Problem: The `CPUID` instruction Function 80000008H (Extended Address Sizes Function) returns the address sizes supported by the processor in the EAX register. This Function returns an incorrect physical address size value of 40 bits. The correct physical address size is 36 bits.

Implication: Function 80000008H returns an incorrect physical address size value of 40 bits.

Workaround: None at this time.

Status: For the steppings affected, see the *Summary Table of Changes*.

S23 Incorrect debug exception (#DB) may occur when a data breakpoint is set on an FP instruction

Problem: The default microcode floating-point event handler routine executes a series of loads to obtain data about the FP instruction that is causing the FP event. If a data breakpoint is set on the instruction causing the FP event, the load in the microcode routine will trigger the data breakpoint resulting in a debug exception.

Implication: An incorrect debug exception (#DB) may occur if data breakpoint is placed on an FP instruction. Intel has not observed this erratum with any commercially available software or system.

Workaround: None at this time.

Status: For the steppings affected, see the *Summary Table of Changes*.

S24 xAPIC may not report some illegal vector errors

Problem: The local xAPIC has an error status register, which records all errors it detects. Bit 6 of this register, the receive Illegal Vector bit, is set when the local xAPIC detects an illegal vector in a message that it receives. When an illegal vector error is received on the same internal clock that the error status register is being written due to a previous error, bit 6 does not get set and illegal vector errors are not flagged.

Implication: The xAPIC may not report some Illegal Vector errors when they occur at approximately the same time as other xAPIC errors. The other xAPIC errors will continue to be reported.

Workaround: None at this time.

Status: For the steppings affected, see the *Summary Table of Changes*.

S25 Enabling no-eviction mode (NEM) may prevent the operation of the second logical processor in a Hyper-Threading Technology enabled boot strap processor (BSP)

Problem: In an HT Technology enabled system, when NEM is enabled by setting Bit 0 of MSR 080h (IA32_BIOS_CACHE_AS_RAM), the second logical processor associated with the BSP may fail to wake up from "Wait-for-SIPI" state.

Implication: In an HT Technology enabled system, the second logical processor associated with the BSP may not respond to SIPI. The OS will continue to operate but with one less logical processor than expected.

Workaround: It is possible for the BIOS to contain a workaround for this erratum.

Status: For the steppings affected, see the *Summary Table of Changes*.

S26 Task priority register (TPR) updates during thermal monitor 2 (TM2), enhanced halt state (CIE), or enhanced Intel SpeedStep® technology (EIST) voltage transitions may cause system hang in DP capable platforms

Problem: Systems with Echo TPR Disable (R/W) bit (bit [23] of IA32_MISC_ENABLE register) set to '0' (default), where xTPR messages are being transmitted on the system bus to the middle agent, may experience system hang during TM2, C1E or EIST transitions.

Implication: This may cause a system hang for the dual-processor capable platforms during TM2, C1E or EIST events.

Workaround: It is possible for the BIOS to contain a workaround for this erratum.

Status: For the steppings affected, see the *Summary Table of Changes*.

S27 Interactions between the instruction translation lookaside buffer (ITLB) and the instruction streaming buffer may cause unpredictable software behavior

Problem: Complex interactions within the instruction fetch/decode unit may make it possible for the processor to execute instructions from an internal streaming buffer containing stale or incorrect information.

Implication: When this erratum occurs, an incorrect instruction stream may be executed resulting in unpredictable software behavior.

Workaround: It is possible for the BIOS to contain a workaround for this erratum.

Status: For the steppings affected, see the *Summary Table of Changes*.

S28 STPCLK# signal assertion under certain conditions may cause a system hang

Problem: The assertion of STPCLK# signal before a logical processor awakens from the "Wait-for-SIPI" state for the first time, may cause a system hang. A processor supporting HT Technology may fail to initialize appropriately, and may not issue a Stop Grant Acknowledge special bus cycle in response to the second STPCLK# assertion.

Implication: When this erratum occurs in an HT Technology enabled system, it may cause a system hang.

Workaround: BIOS should initialize the second thread of the processor supporting HT Technology prior to STPCLK# assertion. Additionally, it is possible for the BIOS to contain a workaround for this erratum.

Status: For the steppings affected, see the *Summary Table of Changes*.

S29 Missing Stop Grant Acknowledge special bus cycle may cause a system hang

Problem: A Stop Grant Acknowledge special bus cycle being deferred by the processor for a period of time long enough for the chipset to deassert and then reassert STPCLK# signal may cause a system hang. A processor supporting HT Technology may fail to detect the deassertion and reassertion of STPCLK# signal, and may not issue a Stop Grant Acknowledge special bus cycle in response to the second STPCLK# assertion.

Implication: When this erratum occurs in an HT Technology enabled system, it may cause a system hang.

Workaround: It is possible for the BIOS to contain a workaround for this erratum.

Status: For the steppings affected, see the *Summary Table of Changes*.

S30 Incorrect duty cycle is chosen when on-demand clock modulation is enabled in a processor supporting Hyper-Threading Technology

Problem: When a processor supporting HT Technology enables on-demand clock modulation on both logical processors, the processor is expected to select the lowest duty cycle of the two potentially different values. When one logical processor enters the AUTOHALT state, the duty cycle implemented should be unaffected by the halted logical processor. Due to this erratum, the duty cycle is incorrectly chosen to be the higher duty cycle of both logical processors.

Implication: Due to this erratum, higher duty cycle may be chosen when the on-demand clock modulation is enabled on both logical processors.

Workaround: None at this time.

Status: For the steppings affected, see the *Summary Table of Changes*.

S31 Memory aliasing of pages as uncacheable memory type and write back (WB) may hang the system

Problem: When a page is being accessed as either UC or write combining (WC) and write back (WB), under certain bus and memory timing conditions, the system may loop in a continual sequence of UC fetch, implicit write back, and RFO retries

Implication: This erratum has not been observed in any commercially available operating system or application. The aliasing of memory regions, a condition necessary for this erratum to occur, is documented as being unsupported in the *IA-32 Intel® Architecture Software Developer's Manual*, Volume 3, Section 10.12.4, Programming the PAT. However, if this erratum occurs the system may hang

Workaround: The pages should not be mapped as either UC or WC and WB at the same time.

Status: For the steppings affected, see the *Summary Table of Changes*.

S32 Using STPCLK# and executing code from very slow memory could lead to a system hang

Problem: The system may hang when the following conditions are met:

1. Periodic STPCLK# mechanism is enabled via the chipset.
2. HT Technology is enabled.
3. One logical processor is waiting for an event (i.e. hardware interrupt).
4. The other logical processor executes code from very slow memory such that every code fetch is deferred long enough for the STPCLK# to be reasserted.

Implication: If this erratum occurs, the processor will go into and out of the sleep state without making forward progress, since the logical processor will not be able to service any pending event. This erratum has not been observed in any commercial platform running commercial software.

Workaround: None at this time.

Status: For the steppings affected, see the *Summary Table of Changes*.

S33 Processor provides a 4-byte store unlock after an 8-byte load lock

Problem: When the processor is in the Page Address Extension (PAE) mode and detects the need to set the Access and/or Dirty bits in the page directory or page table entries, the processor sends an 8 byte load lock onto the system bus. A subsequent 8 byte store unlock is expected, but instead a 4 byte store unlock occurs. Correct data is provided since only the lower bytes change, however external logic monitoring the data transfer may be expecting an 8 byte load lock.

Implication: No known commercially available chipsets are affected by this erratum.

Workaround: None at this time.

Status: For the steppings affected, see the *Summary Table of Changes*.

S34 Machine Check Architecture error reporting and recovery may not work as expected

Problem: When the processor detects errors it should attempt to report and/or recover from the error. In the situations described below, the processor does not report and/or recover from the error(s) as intended.

When a transaction is deferred during the snoop phase and subsequently receives a Hard Failure response, the transaction should be removed from the bus queue so that the processor may proceed. Instead, the transaction is not properly removed from the bus queue, the bus queue is blocked, and the processor will hang.

When a hardware prefetch results in an uncorrectable tag error in the L2 cache, MC0_STATUS.UNCOR and MC0_STATUS.PCC are set but no Machine Check Exception (MCE) is signaled. No data loss or corruption occurs because the data being prefetched has not been used. If the data location with the uncorrectable tag error is subsequently accessed, an MCE will occur. However, upon this MCE, or any other subsequent MCE, .the information for that error will not be logged because MC0_STATUS.UNCOR has already been set and the MCA status registers will not contain information about the error which caused the MCE assertion but instead will contain information about the prefetch error event.

When the reporting of errors is disabled for Machine Check Architecture (MCA) Bank 2 by setting all MC2_CTL register bits to 0, uncorrectable errors should be logged in the IA32_MC2_STATUS register but no machine-check exception should be generated. Uncorrectable loads on bank 2, which would normally be logged in the IA32_MC2_STATUS register, are not logged.

When one-half of a 64-byte instruction fetch from the L2 cache has an uncorrectable error and the other 32-byte half of the same fetch from the L2 cache has a correctable error, the processor will attempt to correct the correctable error but cannot proceed due to the uncorrectable error. When this occurs the processor will hang.

When an L1 cache parity error occurs, the cache controller logic should write the physical address of the data memory location that produced that error into the IA32_MC1_ADDR REGISTER (MC1_ADDR). In some instances of a parity error on a load operation that hits the L1 cache, the

cache controller logic may write the physical address from a subsequent load or store operation into the IA32_MC1_ADDR register.

When an error exists in the tag field of a cache line such that a RFO issued by the processor hits multiple tag fields in the L2 cache (the correct tag and the tag with the error) and the accessed data also has a correctable error, the processor will correctly log the multiple tag match error but will hang when attempting to execute the machine check exception handler.

If a memory access receives a machine check error on both 64 byte halves of a 128-byte L2 cache sector, the IA32_MC0_STATUS register records this event as multiple errors, i.e., the valid error bit and the overflow error bit are both set indicating that a machine check error occurred while the results of a previous error were in the error-reporting bank. The IA32_MC1_STATUS register should also record this event as multiple errors but instead records this event as only one correctable error.

The overflow bit should be set to indicate when more than one error has occurred. The overflow bit being set indicates that more than one error has occurred. Because of this erratum, if any further errors occur, the MCA overflow bit will not be updated, thereby incorrectly indicating only one error has been received.

If an I/O instruction (IN, INS, REP INS, OUT, OUTS, or REP OUTS) is being executed, and if the data for this instruction becomes corrupted, the processor will signal a MCE. If the instruction is directed at a device that is powered down, the processor may also receive an assertion of SMI#. Since MCEs have higher priority, the processor will call the MCE handler, and the SMI# assertion will remain pending. However, while attempting to execute the first instruction of the MCE handler, the SMI# will be recognized and the processor will attempt to execute the SMM handler. If the SMM handler is successfully completed, it will attempt to restart the I/O instruction, but will not have the correct machine state due to the call to the MCE handler. This can lead to failure of the restart and shutdown of the processor.

If PWRGOOD is deasserted during a RESET# assertion causing internal glitches, the MCA registers may latch invalid information.

If RESET# is asserted, then deasserted, and reasserted, before the processor has cleared the MCA registers, then the information in the MCA registers may not be reliable, regardless of the state or state transitions of PWRGOOD.

If MCERR# is asserted by one processor and observed by another processor, the observing processor does not log the assertion of MCERR#. The Machine Check Exception (MCE) handler called upon assertion of MCERR# will not have any way to determine the cause of the MCE.

The Overflow Error bit (bit 62) in the IA32_MC0_STATUS register indicates, when set, that a machine check error occurred while the results of a previous error were still in the error reporting bank (i.e., The Valid bit was set when the new error occurred). If an uncorrectable error is logged in the error-reporting bank and another error occurs, the overflow bit will not be set.

The MCA Error Code field of the IA32_MC0_STATUS register gets written by a different mechanism than the rest of the register. For uncorrectable errors, the other fields in the IA32_MC0_STATUS register are only updated by the first error. Any further errors that are detected will update the MCA Error Code field without updating the rest of the register, thereby leaving the IA32_MC0_STATUS register with stale information.

When a speculative load operation hits the L2 cache and receives a correctable error, the IA32_MC1_Status Register may be updated with incorrect information. The IA32_MC1_Status Register should not be updated for speculative loads.

The processor should only log the address for L1 parity errors in the IA32_MC1_Status register if a valid address is available. If a valid address is not available, the Address Valid bit in the IA32_MC1_Status register should not be set. In instances where an L1 parity error occurs and the address is not available because the linear to physical address translation is not complete or an internal resource conflict has occurred, the Address Valid bit is incorrectly set.

The processor may hang when an instruction code fetch receives a hard failure response from the system bus. This occurs because the bus control logic does not return data to the core, leaving the processor empty. IA32_MC0_STATUS MSR does indicate that a hard fail response occurred.

The processor may hang when the following events occur and the machine check exception is enabled, CR4.MCE=1. A processor that has its STPCLK# pin asserted will internally enter the Stop Grant State and finally issue a Stop Grant Acknowledge special cycle to the bus. If an uncorrectable error is generated during the Stop Grant process it is possible for the Stop Grant special cycle to be issued to the bus before the processor vectors to the machine check handler. Once the chipset receives its last Stop Grant special cycle it is allowed to ignore any bus activity from the processors. As a result, processor accesses to the machine check handler may not be acknowledged, resulting in a processor hang.

Implication: The processor is unable to correctly report and/or recover from certain errors.

Workaround: None at this time.

Status: For the steppings affected, see the *Summary Table of Changes*.

Specification Changes

The Specification Changes listed in this section apply to the following documents:

- *Intel® Xeon™ Processor with 800 MHz System Bus Datasheet* (Document Number 302355)
- *IA-32 Intel® Architecture Software Developer's Manual*, Volume 1: Basic Architecture (Document Number 253665)
- *IA-32 Intel® Architecture Software Developer's Manual*, Volume 2A: Instruction Set Reference, A-M (Document Number 253666)
- *IA-32 Intel® Architecture Software Developer's Manual*, Volume 2B: Instruction Set Reference, N-Z (Document Number 253667)
- *IA-32 Intel® Architecture Software Developer's Manual*, Volume 3: System Programming Guide (Document Number 253668)
- *Intel® Extended Memory 64 Technology Software Developer's Guide*, Volume 1 (Document Number 300834)

  Here is the link: http://developer.intel.com/technology/64bitextensions/300834.htm
- *Intel® Extended Memory 64 Technology Software Developer's Guide*, Volume 2 (Document Number 300835)

  Here is the link: http://developer.intel.com/technology/64bitextensions/300835.htm.

All Specification Changes will be incorporated into a future version of the appropriate Intel Xeon processor documentation.

There are no new Specification Changes for this month.

Specification Clarifications

The Specification Clarifications listed in this section apply to the following documents:

- *Intel® Xeon™ Processor with 800 MHz System Bus Datasheet* (Document Number 302355)
- *IA-32 Intel® Architecture Software Developer's Manual*, Volume 1: Basic Architecture (Document Number 253665)
- *IA-32 Intel® Architecture Software Developer's Manual*, Volume 2A: Instruction Set Reference, A-M (Document Number 253666)
- *IA-32 Intel® Architecture Software Developer's Manual*, Volume 2B: Instruction Set Reference, N-Z (Document Number 253667)
- *IA-32 Intel® Architecture Software Developer's Manual*, Volume 3: System Programming Guide (Document Number 253668)
- *Intel® Extended Memory 64 Technology Software Developer's Guide*, Volume 1 (Document Number 300834)

  Here is the link: http://developer.intel.com/technology/64bitextensions/300834.htm
- *Intel® Extended Memory 64 Technology Software Developer's Guide*, Volume 2 (Document Number 300835)

  Here is the link: http://developer.intel.com/technology/64bitextensions/300835.htm.

All Specification Clarifications will be incorporated into a future version of the appropriate Intel Xeon processor documentation.

There are no new Specification Clarifications for this month.

Documentation Changes

The Documentation Changes listed in this section apply to the following documents:

- *Intel® Xeon™ Processor with 800 MHz System Bus Datasheet* (Document Number 302355)
- *IA-32 Intel® Architecture Software Developer's Manual*, Volume 1: Basic Architecture (Document Number 253665)
- *IA-32 Intel® Architecture Software Developer's Manual*, Volume 2A: Instruction Set Reference, A-M (Document Number 253666)
- *IA-32 Intel® Architecture Software Developer's Manual*, Volume 2B: Instruction Set Reference, N-Z (Document Number 253667)
- *IA-32 Intel® Architecture Software Developer's Manual*, Volume 3: System Programming Guide (Document Number 253668)
- *Intel® Extended Memory 64 Technology Software Developer's Guide*, Volume 1 (Document Number 300834)

  Here is the link: http://developer.intel.com/technology/64bitextensions/300834.htm
- *Intel® Extended Memory 64 Technology Software Developer's Guide*, Volume 2 (Document Number 300835)

  Here is the link: http://developer.intel.com/technology/64bitextensions/300835.htm.

All Documentation Changes will be incorporated into a future version of the appropriate Intel Xeon processor documentation.

There are no new Documentation Changes for this month.